

# Failure Detectors

Poster session: Next week, here

Lots of details in Campus wire post  
but main thing to bear in mind

- What I am looking for 😊
- What would interest/be fun for your classmates

Where we are

RSM → Consensus/Agreement

↳ Validity, agreement, termination

Fail-Stop

Asynch

X

BFT (w/auth.)

X

Partially Synch

✓ if  $n > 2f$

✓ if  $n > 3f+1$

Synch

✓ if  $n > f+1$

✓ if  $n > f+1$

Why gap? Hand-wavy claim

↳ Partial synchrony allows distinguishing

b/w failure & delay

↳ When?

→ For how many nodes?

→ ...

Can we more precisely define what we need?

Why - Theory: Better understand the protocol

Practice: Maintaining/establishing assumptions come at a cost

↳  $n > 2f$

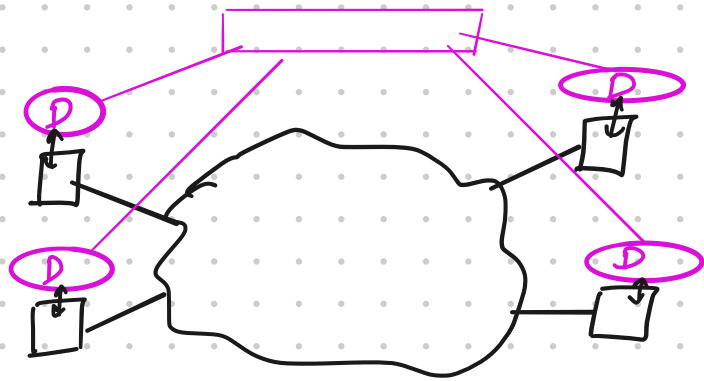
→ Partial synchrony

→ Bounded message latency

•••

Useful to know what is required.

Failure Detectors



## Modeling Notes

-  $F \circ \text{Time} \rightarrow \text{Output}$   
 ( $\mathbb{Z}^+$ ) (can be anything)

In this paper

Output =  $2^\pi$ : Set of processes  
 suspected of failing

Other work

Output =  $\{ \text{Green}, \text{Red} \}$  [Green  $\Rightarrow$  No process  
 is suspected of  
 having failed]

=  $2^\pi$ : Quorum

=  $2^\pi / \text{Green/Red}$  (sufficient for NBAC)

Can we more precisely define what we need?



① Show **consensus** (or  
 other problem) can be



solved if each process had access to  $F$ , assuming some failure model.

Chandra and Toueg:

$$2f+1$$

$$\underbrace{f+1}$$

How many nodes?  $\rightarrow$  Strong ( $\exists$ ) or  $\diamond W(\Omega)$  w/  $2f+1$  processes suffice

② Show that if  $\exists F'$  fodo to solve consensus then one can construct  $F$  from  $F'$

Chandra, Hadzilacos & Toueg (CHT):

$\diamond W(\Omega)$  is *weakest* F.D. for

solving consensus

[Roughly: Use  $F'$  to run a consensus protocol that tracks participants]

$\diamond W \rightarrow \exists p \in \Pi$  that everyone agrees has not failed

$\hookrightarrow$  Leader election?

Chandra & Toueg's Failure Detectors

$F_0$ : time  $\rightarrow$  Set of suspect processes

Accuracy: "What correct processes can be suspected"

- Strong: No correct process is ever suspected

- Weak:  $\exists$  correct process  $p$  that is never suspected.

-  $\diamond$  Strong: Eventually no correct process is suspected

$\exists t$ . s.t.  $t' \geq t \Rightarrow F(t')$  only contains failed processes

-  $\diamond$  Weak:  $\exists$  correct process  $p$  that is eventually not suspected

$\exists p \in \Pi, t: t' \geq t \Rightarrow p \notin F(t')$

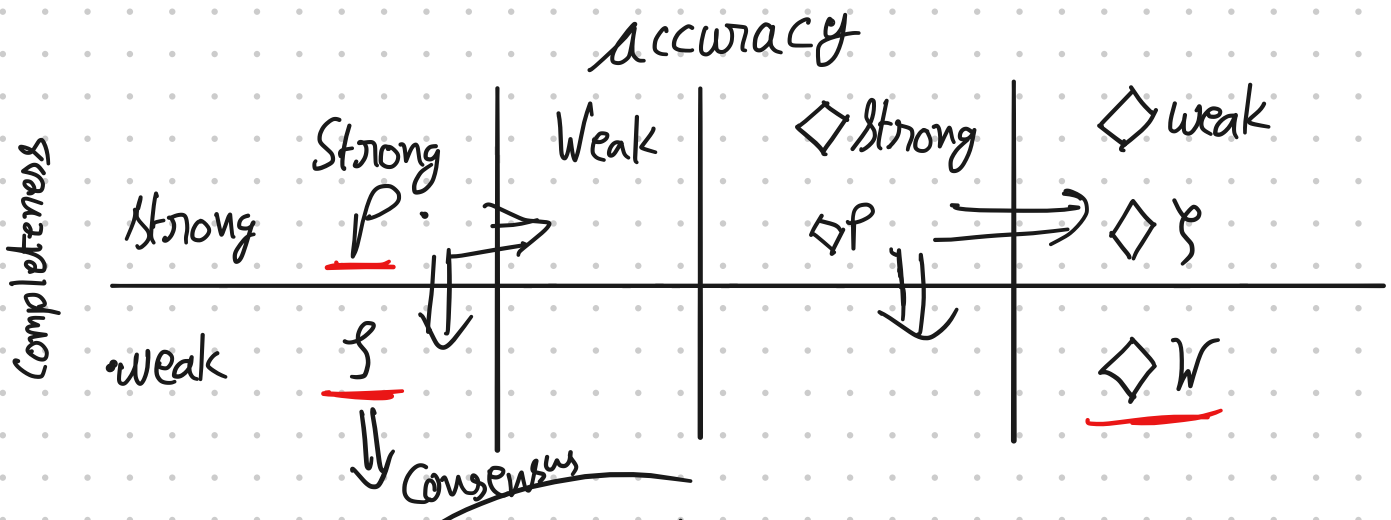
Completeness: "What faulty processes are suspected"

- Strong: Eventually, all failed processes are suspected by all processes



- Weak<sup>o</sup>: Eventually, some correct process suspects each failed process.

Combine these two dimensions to come up with 8 F.D.s



Eight is too many objects.

Observe for

- Comp & accuracy

Strong  $\Rightarrow$  weak

- Accuracy

$\diamond$  Strong  $\Rightarrow$   $\diamond$  weak

Can build F.D. with strong completeness given F.D. with weak completeness.

Weak Comp

$\Rightarrow$  Strong Comp

① Completeness is eventual

↳ Can exchange messages and get FD output from other correct processes

Idea Strong complete output

[all correct processes eventually suspect all failed processes]

||

$\bigcup_{\text{correct } p}$  weak complete output

[a correct process eventually suspects any failed process]

② Tricky: Preserving accuracy.

① Strong  $\leftarrow$  No correct process is suspected

$\Rightarrow$  No processes FD output contains correct processes

$\Rightarrow$  Union does not contain correct  $p$

② Weak  $\leftarrow \exists$  correct  $p$  that is never suspected

[same reasoning as above]

③  $\diamond$  strong  $\wedge$   $\diamond$  weak: Trickier

Union of FD outputs might contain correct

Why? FD output might contain  
 processes (or chosen correct process  
 $p$ ) initially

Need to get rid of it eventually.

How?

Idea 1

while true:

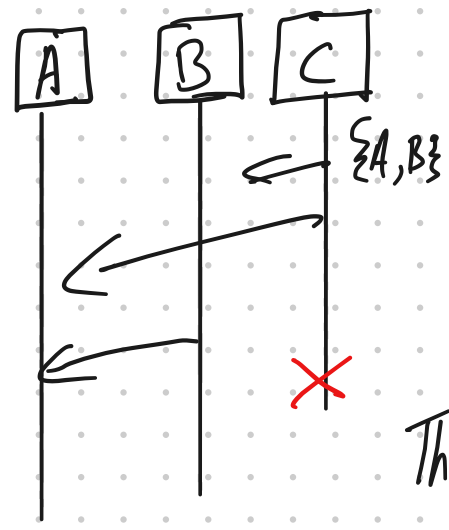
~~send ( $\langle p, \text{fd output} \rangle$ )~~

~~on receive  $\langle p, f \rangle$~~

~~out [ $p$ ]  $\leftarrow f$~~

~~fd output =  $\cup$  out [ $p$ ]~~

This doesn't work. Why?



Idea 2:

Every process  $p$  executes the following:

$output_p \leftarrow \emptyset$

**cobegin**

|| Task 1: repeat forever

{  $p$  queries its local failure detector module  $\mathcal{D}_p$   
 $suspects_p \leftarrow \mathcal{D}_p$   
 send ( $p, suspects_p$ ) to all }

} Send FD output

|| Task 2: when receive ( $q, suspects_q$ ) for some  $q$

$output_p \leftarrow (output_p \cup suspects_q) - \{q\}$

}

**coend**

Why does this work?

Accuracy

C O M P	Strong P	Weak S	◇ Strong ◇ P	◇ weak ◇ S
	<u>weak</u> S	W	◇ Q	◇ W

Can focus on one or the other.

Two results

- ①  $S$  (on  $W$ )  $\rightarrow$  Consensus with  $f+1$  nodes
- ②  $\diamond S$  (on  $\diamond W$ )  $\rightarrow$  Consensus with  $2f+1$  nodes.

Going to start with ②  $\leftarrow$  Similar to what we have seen before.

Core idea

① Nodes take it in turn to be leader

$\hookrightarrow$  When a node becomes leader it tries to replicate a value (proposal) to all correct nodes.

→ Leader counts how many times its value has been replicated & decides (commits) when the value is sufficiently replicated  
 ↳ Informs nodes about decisions

② Other processes move onto another leader if F.D. suspects current leader

Timing might lead to a scenario where current leader is suspected after value is sufficiently replicated (or potentially committed)

Must ensure that a sufficiently replicated value is used for all future proposals.

↳ Compare to Paxos

V1  
 (P) ✓

V2  
 (P) No next ✓  
 S: strong acc strong acc

P2. If a proposal with value  $v$  is chosen, then every higher-numbered proposal that is chosen has value  $v$ .

Use similar idea ← Quorum Intersection

to decide proposal.

Every process  $p$  executes the following:

procedure  $propose(v_p)$

$estimate_p \leftarrow v_p$  {  $estimate_p$  is  $p$ 's estimate of the decision value }  
 $state_p \leftarrow undecided$   
 $r_p \leftarrow 0$  {  $r_p$  is  $p$ 's current round number }  
 $ts_p \leftarrow 0$  {  $ts_p$  is the last round in which  $p$  updated  $estimate_p$ , initially 0 }

{ Rotate through coordinators until decision is reached }

while  $state_p = undecided$

$r_p \leftarrow r_p + 1$   
 $c_p \leftarrow (r_p \bmod n) + 1$  {  $c_p$  is the current coordinator }

Phase 1: { All processes  $p$  send  $estimate_p$  to the current coordinator }

send  $(p, r_p, estimate_p, ts_p)$  to  $c_p$

Phase 2: { The current coordinator gathers  $\lceil \frac{n+1}{2} \rceil$  estimates and proposes a new estimate }

if  $p = c_p$  then

wait until [for  $\lceil \frac{n+1}{2} \rceil$  processes  $q$ : received  $(q, r_p, estimate_q, ts_q)$  from  $q$ ]  
 $msgs_p[r_p] \leftarrow \{(q, r_p, estimate_q, ts_q) \mid p \text{ received } (q, r_p, estimate_q, ts_q) \text{ from } q\}$   
 $t \leftarrow$  largest  $ts_q$  such that  $(q, r_p, estimate_q, ts_q) \in msgs_p[r_p]$   
 $estimate_p \leftarrow$  select one  $estimate_q$  such that  $(q, r_p, estimate_q, t) \in msgs_p[r_p]$   
send  $(p, r_p, estimate_p)$  to all

Phase 3: { All processes wait for the new estimate proposed by the current coordinator }

wait until [received  $(c_p, r_p, estimate_{c_p})$  from  $c_p$  or  $c_p \in \mathcal{D}_p$ ]{ Query the failure detector }

if [received  $(c_p, r_p, estimate_{c_p})$  from  $c_p$ ] then {  $p$  received  $estimate_{c_p}$  from  $c_p$  }

$estimate_p \leftarrow estimate_{c_p}$

$ts_p \leftarrow r_p$

send  $(p, r_p, ack)$  to  $c_p$

else send  $(p, r_p, nack)$  to  $c_p$

{  $p$  suspects that  $c_p$  crashed }

Phase 4: { The current coordinator waits for  $\lceil \frac{n+1}{2} \rceil$  replies. If they indicate that  $\lceil \frac{n+1}{2} \rceil$  processes adopted its estimate, the coordinator R-broadcasts a decide message }

if  $p = c_p$  then

wait until [for  $\lceil \frac{n+1}{2} \rceil$  processes  $q$ : received  $(q, r_p, ack)$  or  $(q, r_p, nack)$ ]

if [for  $\lceil \frac{n+1}{2} \rceil$  processes  $q$ : received  $(q, r_p, ack)$ ] then

R-broadcast  $(p, r_p, estimate_p, decide)$

{ If  $p$  R-delivers a decide message,  $p$  decides accordingly }

when R-deliver  $(q, r_q, estimate_q, decide)$

if  $state_p = undecided$  then

decide  $(estimate_q)$

$state_p \leftarrow decided$

Phase 1

Phase 2

Count & Commit.

Termination? Eventually correct  $p$ , who no one suspects will become leader.

①  $S \text{ (con } W) \rightarrow$  Consensus with  $f+1$  nodes

Why gap:  $\exists$  correct process  $p$  that is never suspected

$\hookrightarrow$  Have correct process relay

Information from other processes

Challenge: Don't know identity of correct  $p$  ahead of time

↳ Agreeing on correct  $p$  equivalent to consensus

so instead rely on multiple rounds of communication

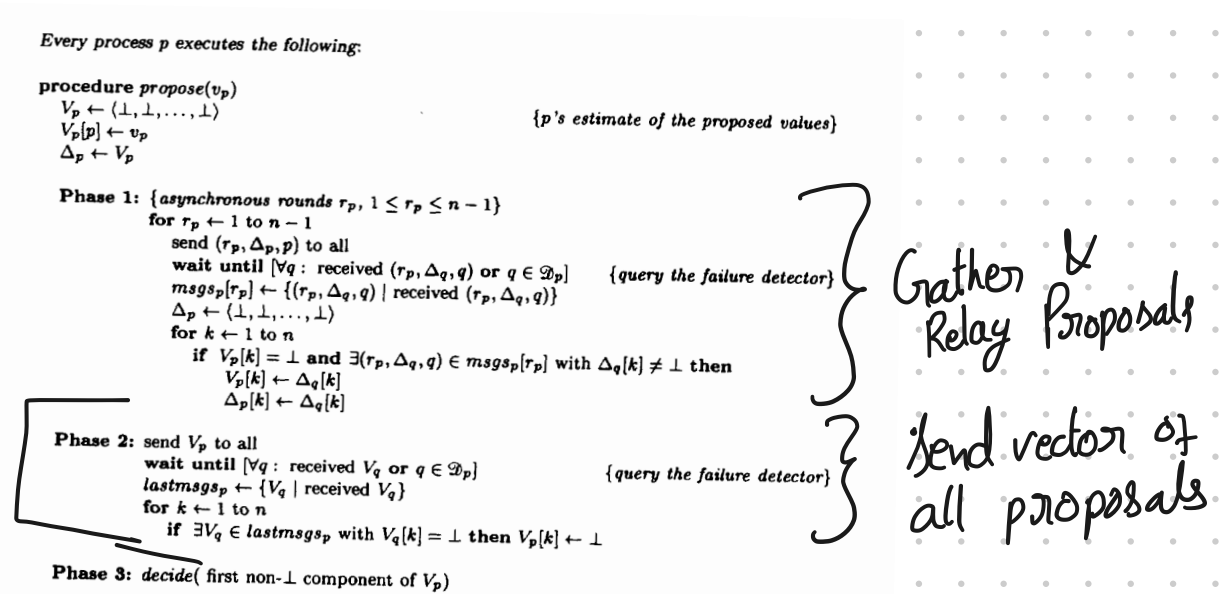
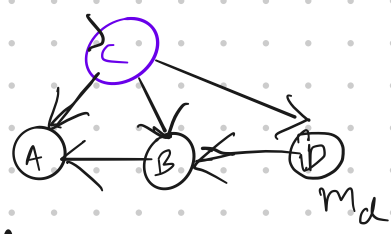


FIG. 5. Solving Consensus using any  $\mathcal{D} \in \mathcal{F}$ .

### Failure Detectors in practice

- Desirable but
- Accuracy is hard



↳ Weak: How to ensure that some process  $p$  is never suspected?

↳ How to choose  $p$ ?

↳ Strong: No correct process is even suspected

↳ Delay vs failure

↳ Prior work (Falcon, others)

↳ Kill suspected nodes

↳ Likely to violate any assumptions about # of process failures.

Pigeon

↳ Provide information about failure certainty (Warnings vs facts)

→ Use additional sources of information to improve predictions } Core Benefit!